# 拥抱开源：小米的经验分享


崔宝秋

小米首席架构师


2014.4.27

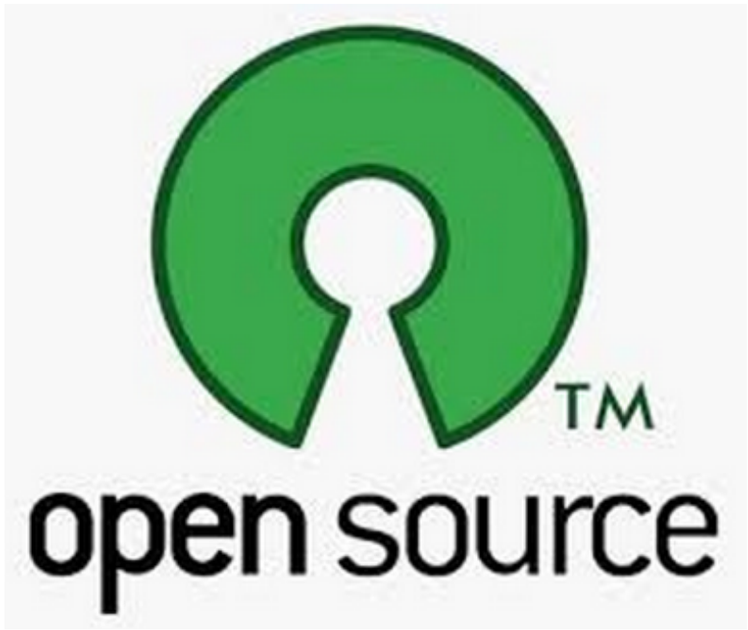
VISIT…

# 提纲

- 拥抱开源：小米团队文化的一部分
- 小米开源的现状
- 小米经验分享
- Q & A

# 为什么要拥抱开源

- 所有的互联网创业公司都离不开开源
  - 站在巨人肩膀上，快速创新
- 共享技术，回报社区
- 有助于吸引人才
- 有助于吸引外来贡献
- 有助于提高软件质量



# 使用开源的一些顾虑

- 学习成本
- 缺乏掌控
- 不够灵活
- 知识产权

# 自主研发的一些问题

- 研发成本大：很可能不如花时间掌握开源软件
- 有试错风险：无前车之鉴
- 常见错误：
  - 早期性能和功能评估的误判
  - 低估随业务增长带来的压力

# 小米在开源上的原则

快

不重造轮子

不用则已，要用则精

永抱开放和共享的态度

极力推出自己的committer

# 为什么要推出自己的Committer

- 不仅仅要站在巨人的肩膀上，更要为巨人指方向
- 赢得话语权，影响开源项目的走向
  - 向有利于自己公司的方向发展
- 避免内部分支与社区主干分支渐行渐远
  - 否则，有可能远离社区，失去使用开源的意义

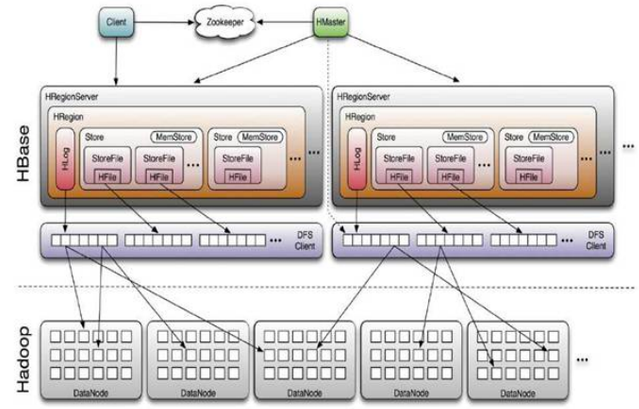
# 小米使用的开源软件

| 日志框架 | 服务框架 | HTTP Web 框架 | 消息队列 |
| --- | --- | --- | --- |
| Thrift<br>Scribe | Thrift<br>ZooKeeper | Rose<br>Nginx | Kafka<br>RabbitMQ |

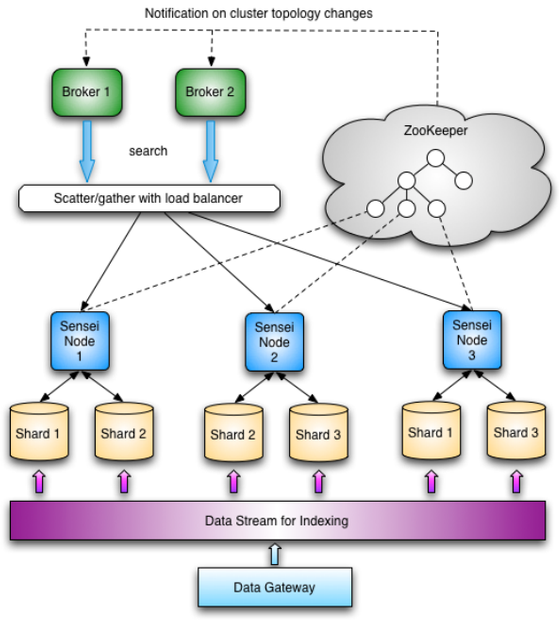
| 分布式缓存 | 存储业务 | 监控报警 | 数据处理 |
| --- | --- | --- | --- |
| Memcached | HBase<br>MySQL<br>Riak<br>Voldemort | Ganglia<br>Nagios | Hadoop<br>Storm<br>Hive<br>Spark<br>Azkaban |

# 小米参与开发的开源软件

HBase




SenseiDB



# 小米自己开源的软件

MIUI 系列工具

Minos：分布式部署和监控工具

Chronos：高可用Timestamp服务

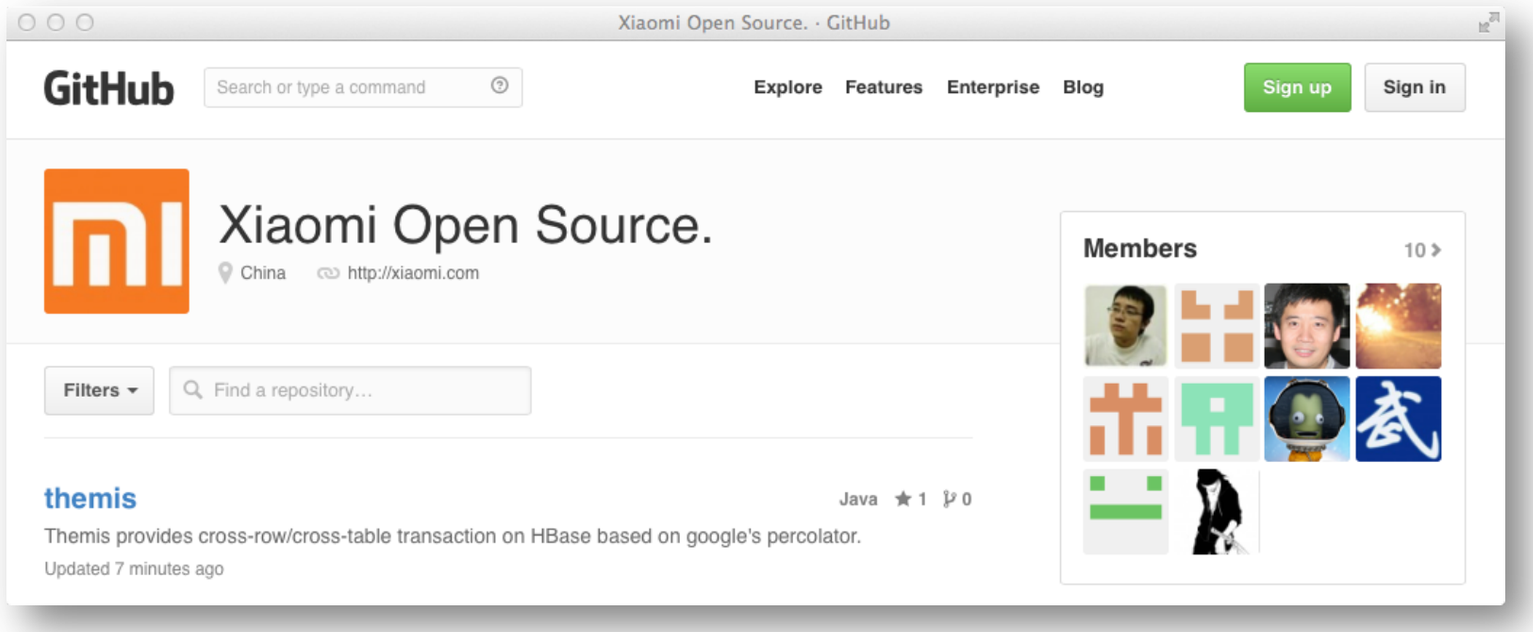
Themis：HBase 跨行跨表事务的实现

其他（包括一些运维工具等）

# 小米开源软件的两个入口

- https://github.com/micode

- https://github.com/XiaoMi



# 1. 开源软件的选型

- **必须能满足业务需求**
  - 深度调研、测试、比较
  - 最好能踏着前人的脚印
- **必须能驾驭**
  - 源代码级别上的深入了解
  - 足够强的问题诊断、解决能力

# 1. 开源软件的选型：案例

- HBase
  - Facebook类似场景和用户
  - 我们的技术积累
- Voldemort
  - 之前的使用经验
  - 离线计算结果无缝推送到线上业务的需求

# 2. 平衡在开源社区和公司业务上的投入

# 2. 平衡在开源社区和公司业务上的投入：案例

- HBase
  - 早期团队无业务压力
  - 给团队足够的时间来学习、掌握HBase
  - 无业务压力时鼓励向社区提交各类patch
  - 有业务压力时以业务需求为第一优先级

# 3. 业务驱动

- 不凭空提改动
- 不凭空提需求
- 尽量解决与公司业务相关的问题

# 3. 业务驱动：案例

- HBase的反向扫描（Reverse Scan）
  - **业务需求**：米聊大群消息全存储，往前/往后浏览
  - 社区中有此需求一年以上，但公认很难，无下文
  - 2-3天想到解决方案，3-4天实现主要代码，之后经数周调试
  - 提交到社区，被接收，成为HBase 0.98.0四大改进之一
- HBase的Delete Family Version
  - **业务需求**：短信搜索，需要删除Column Family中某版本的所有项
  - 开会讨论时，当场判断应该可以实现，设计和代码一天完成
  - 提交到社区，被接收

# 4. 如何更好地回报社区

积极交流
参与讨论

清晰地描述自己
的想法和算法

坚持自己
正确的观点

为自己提交的
代码负责

# 4. 如何更好地回报社区：工程师重要素质

技术能力

交流能力

书面和英文

热爱开源

责任心

长期投入

Commitment

# 4. 如何更好地回报社区：案例

## HBase写性能的优化

- 工作中发现HBase写性能不理想
- 3 – 4周实现原型，提高性能3 – 4倍
- 提交社区，被质疑，多次反复交流
- 近一年后终于被接受，写性能在压力大时提高3.5倍，成为HBase 0.98.0 四大改进中的第一大改进

# 5. 如何推出自己的Committer

- 提高自己的技术水平：打铁还需自身硬
- 提交足够多的patch（质和量兼顾）
- 保持在开源社区内的长期活跃度
- 积极与开源项目的负责人和重要committer们交流
- 向社区展示团队的整体力量和公司的投入
- 梯队式推进

# 5. 如何推出自己的Committer：案例

- 在一年多时间里小米HBase团队
  - 提交了228个改动（130个被接受）
  - 推出了两个committer

# Q & A

# 谢谢！

cuibaoqiu@xiaomi.com